5 PROTOCOLO HTTP

En este capítulo vamos a conocer la forma de enviar un mensaje SOAP a través de la red. La especificación SOAP no indica ninguna manera específica de transportar la información, de modo que los mensajes podrían viajar a través de protocolos de transporte, archivos de texto o cualquier otro método de transferencia de datos.

El modelo TCP/IP cuenta con diversos protocolos en su capa de aplicación: HTTP, SMTP y FTP son tres de los más importantes. En principio, cualquiera de ellos puede ser utilizado para la transferencia de mensajes SOAP. En este proyecto utilizaremos HTTP, el protocolo estándar para la web y el más usado en los servicios web XML.

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes web y los servidores HTTP. La especificación completa del protocolo HTTP/1.0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web. Actualmente la versión más reciente de HTTP es la 1.1, y su especificación se encuentra recogida en el documento RFC 2616.

5.1 Características y funcionamiento

Desde el punto de vista de las comunicaciones, HTTP se establece sobre la capa de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto web es identificado por su URL.

Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres US-ASCII de 7 bits.

- Permite la transferencia de objetos multimedia, codificando los archivos binarios en cadenas de caracteres. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.

- Existen ocho verbos que permiten que un cliente pueda dialogar con el servidor. Los tres más utilizados son: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD, para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).

- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto. Con la versión HTTP 1.1 se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado HTTP Keep Alive, es empleado por la mayoría de los clientes y servidores modernos.

- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.

- Cada objeto al que se aplican los verbos del protocolo está identificado a través de un localizador uniforme de recurso (URL) único.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el navegador.

2. El cliente web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el puerto (de carácter opcional; el valor por defecto es 80) y el objeto requerido del servidor.

3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.

4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor, etc.

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.

6. Se cierra la conexión TCP. Si no se utiliza el modo HTTP Keep Alive, este proceso se repite para cada acceso al servidor HTTP.

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Solo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta.

La estructura general de los dos tipos de mensajes se puede ver en el siguiente esquema:

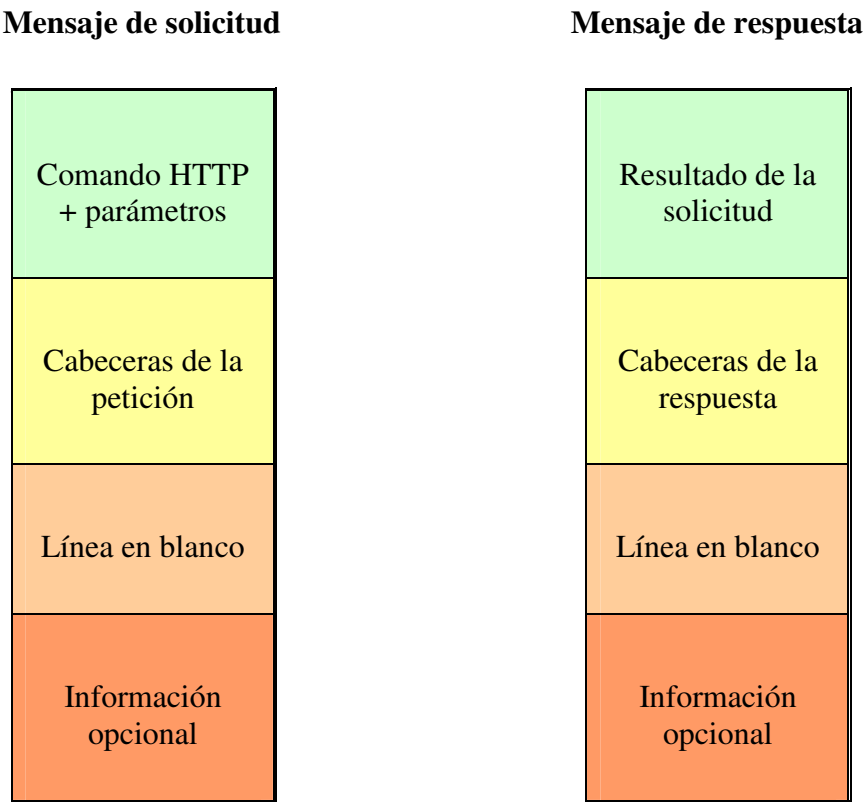

Figura 5.1: Estructura de los mensajes HTTP

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que la primera línea de la respuesta contiene el resultado de la operación identificado por un código numérico que permite conocer el éxito o fracaso de dicha operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales) que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con un par CR-LF (retorno de carro más nueva línea). El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor.

A continuación se muestran los mensajes intercambiados por cliente y servidor cuando accedemos a la dirección web http://portal.us.es con el navegador Mozilla Firefox. Cada color de fondo se identifica con la estructura de los mensajes anteriormente explicada.

Petición del cliente:

```
GET /index.htm HTTP/1.1
```

```
Host: portal.us.es:80
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES;
      rv:1.8.0.6) Gecko/20060728 Firefox/1.5.0.6
Accept: text/xml,application/xml,application/xhtml+xml,text/html;
q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: es,es-es;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Respuesta del servidor:

```
HTTP/1.0 200 OK
```

```
Server: Zope/(Zope 2.8.6-final, python 2.3.5, linux2) ZServer/1.1
Plone/Unknown
Date: Wed, 09 Aug 2006 17:45:16 GMT
Content-Length: 22496
Content-Language:
X-Cache-Headers-Set-By: CachingPolicyManager:
/us/caching_policy_manager
Expires: Wed, 09 Aug 2006 17:55:16 GMT
Cache-Control: max-age=600
Content-Type: text/html;charset=utf-8
Age: 198
X-Cache: HIT from vega2.us.es
X-Cache-Lookup: HIT from vega2.us.es:80
Connection: keep-alive
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
...
</html>
```

5.2 Comandos de HTTP

El protocolo HTTP/1.1 consta de los siguientes comandos:

- **GET**: Sirve para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento ubicado en la dirección especificada por dicha URL.

- **HEAD**: Es un comando similar a GET pero que pide solamente la cabecera del objeto. Lo utilizan principalmente los gestores de cachés de páginas o los servidores proxy para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.

- **POST**: Este comando envía datos de información al servidor, normalmente procedentes de un formulario web, para que el servidor los administre o los añada a una base de datos.

- **PUT**: Almacena un objeto en la URL especificada. Si la dirección de destino ya contenía un objeto, se considera que se está enviando una versión actualizada del mismo.

- **DELETE**: Elimina el objeto especificado. Este comando es muy poco utilizado.

- **TRACE**: Realiza un eco de la solicitud recibida para que el cliente pueda conocer qué servidores intermedios están añadiendo información o modificando la petición.

- **OPTIONS**: Devuelve los métodos HTTP que soporta el cliente. Se suele utilizar para comprobar la funcionalidad de un servidor web.

- **CONNECT**: Se utiliza en los servidores proxy que puedan establecer un túnel dinámicamente (por ejemplo, un túnel SSL).

Ante cada transacción con un servidor HTTP, éste devuelve un código numérico en la primera línea del mensaje de respuesta que informa sobre el resultado de la operación. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error.

Los códigos de estado están clasificados en cinco categorías.

- 1xx: Mensajes informativos.

- 2xx: Mensajes asociados con operaciones realizadas correctamente.

- 3xx: Mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación.

- 4xx: Errores del cliente; el requerimiento contiene algún error, o no puede ser realizado.

- 5xx: Errores del servidor, que no ha podido llevar a cabo una solicitud.

En la siguiente tabla podemos ver una lista con los códigos que se utilizan con mayor frecuencia:

Código	*Comentario*	*Descripción*

200	OK	Operación realizada satisfactoriamente.
201	Created	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. Este nuevo objeto ya está disponible.
202	Accepted	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. El nuevo objeto no está disponible por el momento. En el cuerpo de la respuesta se debe informar sobre la disponibilidad de la información.
204	No Content	La operación ha sido aceptada, pero no ha producido ningún resultado de interés. El cliente no deberá modificar el documento que está mostrando en este momento.
301	Moved Permanently	El objeto al que se accede ha sido movido a otro lugar de forma permanente. El servidor proporciona, además, la nueva URL en el campo Location de la respuesta.
302	Found	El objeto al que se accede ha sido movido a otro lugar de forma temporal. El servidor proporciona, además, la nueva URL en el campo Location de la respuesta. El cliente no debe modificar ninguna de las referencias a la URL errónea.
304	Not Modified	Se devuelve cuando se hace un GET condicional y el documento no ha sido modificado

400	Bad Request	La petición tiene un error de sintaxis y no es entendida por el servidor.
401	Unauthorized	La petición requiere una autorización especial, que normalmente consiste en un nombre y clave que el servidor verificará. El campo WWW-Autenticate informa de los protocolos de autentificación aceptados para este recurso.
403	Forbidden	Está prohibido el acceso a este recurso. No es posible utilizar una clave para modificar la protección.
404	Not Found	La URL solicitada no existe.
500	Internal Server Error	El servidor ha tenido un error interno, y no puede continuar con el procesamiento.
501	Not Implemented	El servidor no tiene capacidad, por su diseño interno, para llevar a cabo el requerimiento del cliente.
502	Bad Gateway	El servidor, que está actuando como proxy o pasarela, ha encontrado un error al acceder al recurso que había solicitado el cliente.
503	Service Unavailable	El servidor está actualmente deshabilitado y no es capaz de atender el requerimiento.

Tabla 5.1: Lista de códigos HTTP

5.3 Codificación de la información

Una de las características de HTTP reside en que la comunicación entre cliente y servidor se realiza a partir de caracteres US-ASCII de 7 bits. El problema aparece cuando deseamos realizar la transmisión de un archivo binario, cuyo contenido no puede ser representado por este grupo tan reducido de caracteres.

Para solventar esta limitación se utiliza el estándar de Internet MIME (Extensiones de correo de Internet multipropósito). Son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una parte importante de MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos. La especificación de este estándar se encuentra recogidas en las RFC 2045, 2046, 2047, 2048 y 2049.

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME, que se especifica en el campo de cabecera "Content-Type". De esta forma, el protocolo puede intercambiar cualquier tipo de dato sin preocuparse de su contenido. La identificación MIME permitirá que el receptor trate adecuadamente los datos. Existen nueve tipos definidos por la IANA: application, audio, example, image, message, model, multipart, text y video. Dentro de cada tipo existen multitud de subtipos: (text/html, image/gif, image/jpeg, audio/x-mpeg, video/quicktime…).

Existen tres métodos básicos de codificación:

- **7 bit**: Utilizado por defecto, supone que los archivos son de texto.

- **Quoted-printable**: Se usa para codificar texto con caracteres que excedan de los 7 bits utilizados en US-ASCII. Con esto podremos representar caracteres de otros alfabetos como los idiomas procedentes del latín. Se codifica en 3 caracteres de 7 bits. Por ejemplo, la letra "ñ" se corresponderá con los caracteres "=F1".

- **Base64**: Se utiliza para codificar contenido no legible por humanos, como por ejemplo archivos multimedia. Cada 3 octetos se codifican en 4 caracteres que pertenecen a un subconjunto de 64 caracteres imprimibles del US-ASCII. Los pasos para codificar un conjunto de octetos en caracteres codificados en Base64 son:

    1. Se agrupan todos los bits, quedándose al principio el bit más significativo y al final el menos significativo.

2. Se toman grupos de 6 bits. Si en el último grupo quedan menos de 6 bits, se rellena con ceros.

3. Obtenemos el valor decimal de cada uno de los grupos.

4. Identificamos cada valor con un carácter de la tabla adjunta:

0 A	17 R	34 i	51 z
1 B	18 S	35 j	52 0
2 C	19 T	36 k	53 1
3 D	20 U	37 l	54 2
4 E	21 V	38 m	55 3
5 F	22 W	39 n	56 4
6 G	23 X	40 o	57 5
7 H	24 Y	41 p	58 6
8 I	25 Z	42 q	59 7
9 J	26 a	43 r	60 8
10 K	27 b	44 s	61 9
11 L	28 c	45 t	62 +
12 M	29 d	46 u	63 /
13 N	30 e	47 v	
14 O	31 f	48 w	(pad) =
15 P	32 g	49 x	
16 Q	33 h	50 y	

5. Se añaden al final tantos signos "=" como parejas de ceros hayamos añadido en el paso 2.

Para comprender mejor estos pasos vamos a ilustrar el proceso de codificación con un sencillo ejemplo:

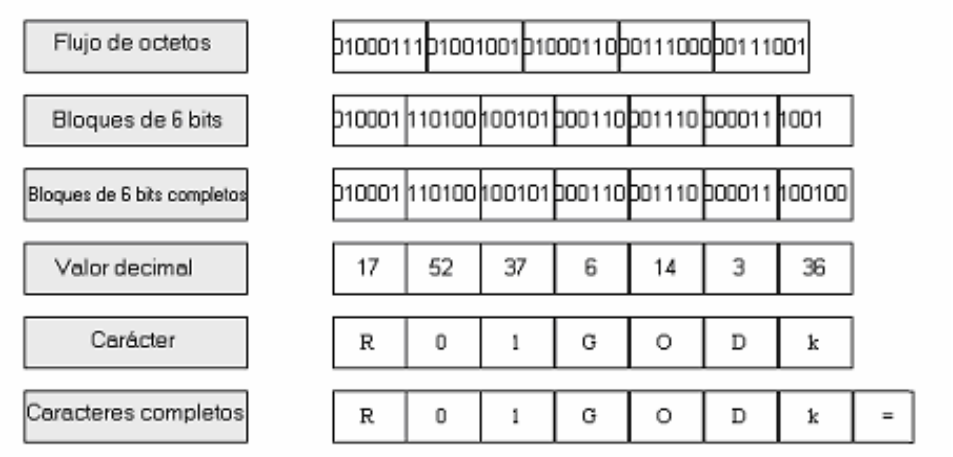

Figura 5.2: Ejemplo de codificación en Base64

5.4 Consideraciones finales

El protocolo cliente-servidor HTTP es el estándar en la web y el más utilizado en los Servicios Web XML. Sus características hacen que haya sido el protocolo preferido por los servidores web XML para realizar el intercambio de peticiones.

En primer lugar hemos comentado el funcionamiento básico del protocolo, y se ha mostrado la estructura de los mensajes de petición y respuesta. A continuación se han presentado los ocho comandos que se utilizan en las peticiones HTTP para facilitar la comunicación del cliente con el servidor.

Por último hemos hablado brevemente de MIME, el estándar que permite enviar dentro de HTTP información que no es de tipo texto. Tras enumerar los tres métodos básicos de codificación: 7 bit, Quoted-printable y Base64, nos hemos detenido en este último para detallar su funcionamiento, ya que nos basaremos en él para codificar las imágenes que se van a transmitir.